

Lab Exercise 4– Terraform Variables

Objective:

Learn how to define and use variables in Terraform configuration.

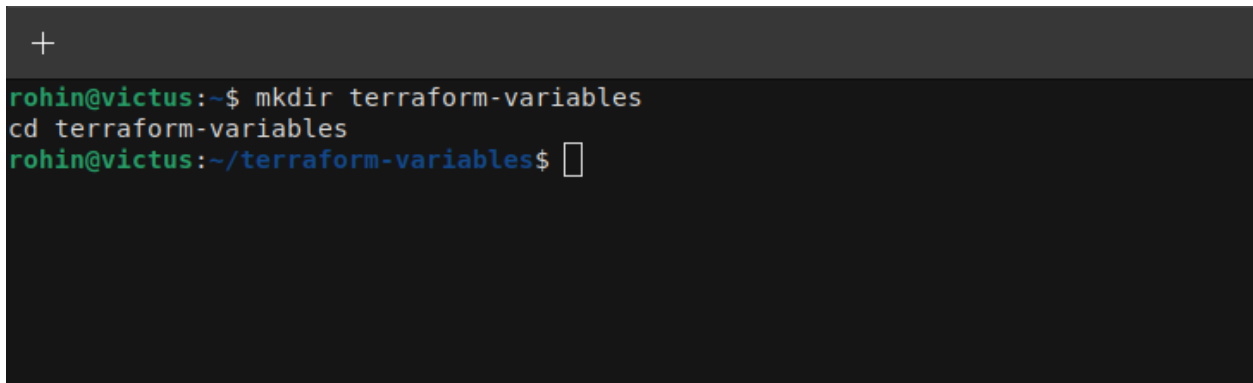
Prerequisites:

- Install Terraform on your machine.

Steps:

1. Create a Terraform Directory:

- Create a new directory for your Terraform project.

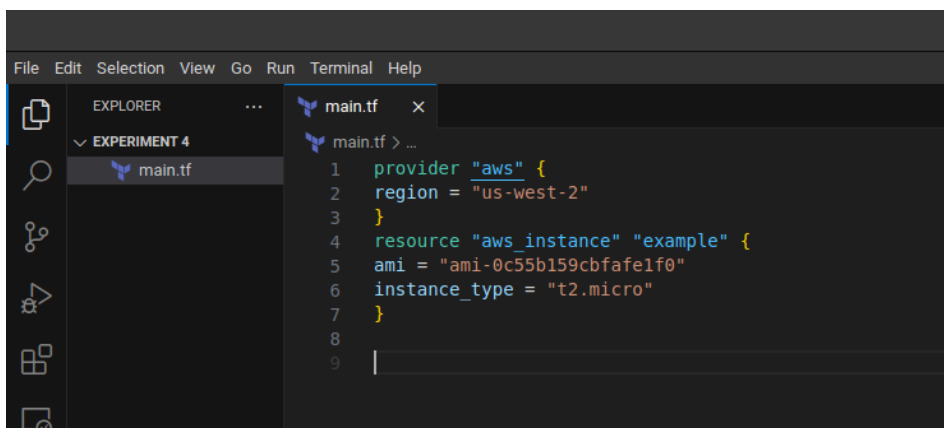
A terminal window with a dark background. The prompt is 'rohin@victus:~\$'. The user enters 'mkdir terraform-variables' and presses enter. The prompt changes to 'rohin@victus:~/terraform-variables\$'. The user enters 'cd terraform-variables' and presses enter. The prompt changes to 'rohin@victus:~/terraform-variables\$'.

```
+
rohin@victus:~$ mkdir terraform-variables
cd terraform-variables
rohin@victus:~/terraform-variables$
```

2. Create a Terraform Configuration File:

- Create a file named main.tf within your project directory.

main.tf

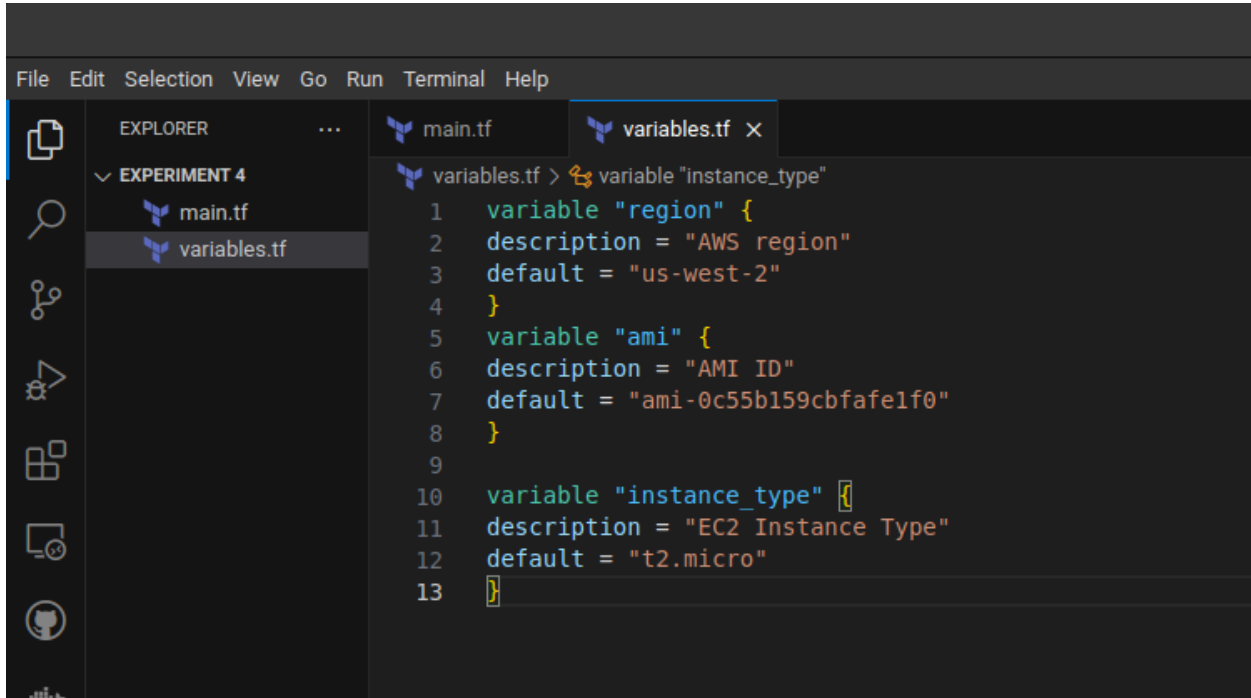
A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a folder named 'EXPERIMENT 4' containing a file named 'main.tf'. The main editor area shows the content of 'main.tf'. The code defines an AWS provider and an AWS instance resource.

```
File Edit Selection View Go Run Terminal Help
EXPLORER
EXPERIMENT 4
main.tf
main.tf
1 provider "aws" {
2   region = "us-west-2"
3 }
4 resource "aws_instance" "example" {
5   ami = "ami-0c55b159cbfafe1f0"
6   instance_type = "t2.micro"
7 }
8
9
```

3. Define Variables:

- Open a new file named variables.tf. Define variables for region, ami, and Instance_type.

variables.tf



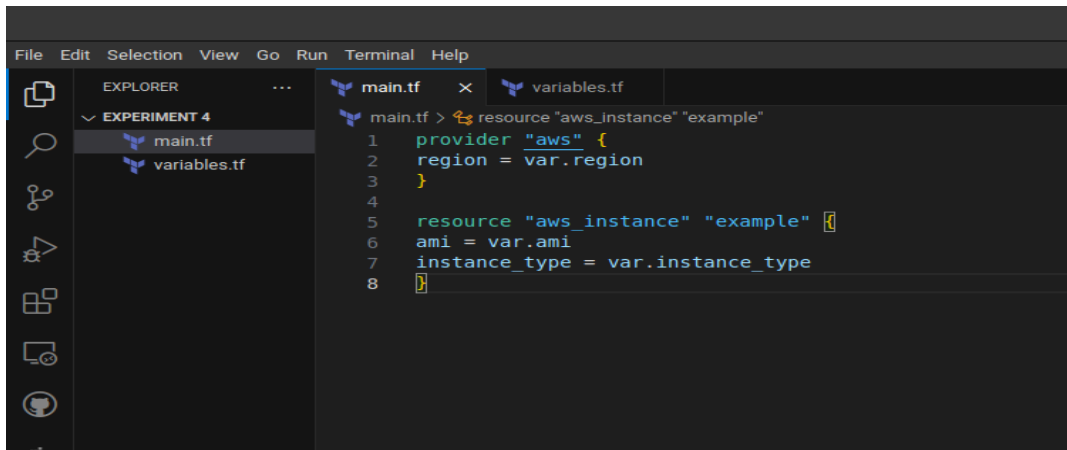
The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. Under 'EXPERIMENT 4', there are two files: 'main.tf' and 'variables.tf'. The 'variables.tf' file is selected and its content is displayed in the main editor area. The code defines three variables: 'region', 'ami', and 'instance_type'.

```
File Edit Selection View Go Run Terminal Help
EXPLORER
EXPERIMENT 4
  main.tf
  variables.tf
main.tf variables.tf X
variables.tf > variable "instance_type"
1  variable "region" {
2    description = "AWS region"
3    default = "us-west-2"
4  }
5  variable "ami" {
6    description = "AMI ID"
7    default = "ami-0c55b159cbfaffe1f0"
8  }
9
10 variable "instance_type" {
11   description = "EC2 Instance Type"
12   default = "t2.micro"
13 }
```

4. Use Variables in main.tf:

- Modify main.tf to use the variables.

main.tf



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. Under 'EXPERIMENT 4', there are two files: 'main.tf' and 'variables.tf'. The 'main.tf' file is selected and its content is displayed in the main editor area. The code defines an AWS instance resource using the variables defined in 'variables.tf'.

```
File Edit Selection View Go Run Terminal Help
EXPLORER
EXPERIMENT 4
  main.tf
  variables.tf
main.tf variables.tf X
main.tf > resource "aws_instance" "example"
1  provider "aws" {
2    region = var.region
3  }
4
5  resource "aws_instance" "example" {
6    ami = var.ami
7    instance_type = var.instance_type
8  }
```

5. Initialize and Apply:

•Run the following Terraform commands to initialize and apply the configuration.

```
rohin@victus:~/UPES/Sem_6/Experiment 4$ terraform apply

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

# aws_instance.example will be created
+ resource "aws_instance" "example" {
  + ami                  = "ami-03f4878755434977f"
  + arn                  = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone     = (known after apply)
  + cpu_core_count        = (known after apply)
  + cpu_threads_per_core   = (known after apply)
  + disable_api_stop       = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized          = (known after apply)
  + get_password_data      = false
  + host_id               = (known after apply)
  + host_resource_group_arn = (known after apply)
  + iam_instance_profile   = (known after apply)
  + instance_type          = "t3.micro"
  + key_name               = (known after apply)
  + monitoring              = false
  + network_interface_ids  = []
  + placement_group        = (known after apply)
  + primary_interface       = (known after apply)
  + subnet_id              = "subnet-0a1b2c3d"
  + tags                   = {}
  + user_data               = ""
  + vpc_security_group_ids = ["sg-0a1b2c3d"]
}
```

Observe how the region changes based on the variable override.

6. Clean Up:

After testing, you can clean up resources.

```
`terraform destroy`
```

Confirm the destruction by typing yes.

7. Conclusion:

This lab exercise introduces you to Terraform variables and demonstrates how to use them in your configurations. Experiment with different variable values and overrides to understand their impact on the infrastructure provisioning process.